Sep payable algorithms for distance transformations on irregular grids

Antoine Vacavant a,*, David Coeurjolly b, c, Laure Tougne b, d

a Clermont Université, Université d’Auvergne, ISIT F-63000, France
b Université Lyon 1, LIRIS, UMR5205 F-69622, France

c Université Lyon 2, LIRIS, UMR5205 F-69676, France

Available online 24 November 2010

ARTICLE INFO

Article history:
Available online 24 November 2010

Keywords:
Squared Euclidean distance transformation
Irregular grids
Quadtree
Voronoi diagrams

ABSTRACT

In this article, we propose to investigate two extensions of the E²DT (squared Euclidean Distance Transformation) on irregular isothetic grids (or i-grids), such as quadtree/octree or run-length encoded d-dimensional images. We enumerate the advantages and drawbacks of the i-CDT, based on the cell centres, and the one of the i-BDT, which uses the cell borders. One of the main problem we mention is that no efficient algorithm has been designed to compute both transforms in arbitrary dimensions. To tackle this problem, we describe in this paper two algorithms, separable in dimension, to compute these distance transformations in the two-dimensional case, and we show that they can be easily extended to higher dimensions.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The distance transformation (Klette and Rosenfeld, 2004; Rosenfeld and Pfaltz, 1968) (DT) of a binary image consists in labeling each point of a discrete object E (i.e. foreground) with its shortest distance to the complement of E (i.e. background). This process has been widely studied and developed on regular grids (see Fabbri et al., 2008 for a survey). Some specific extensions of the DT to non-regular grids also exist, such as elongated grids (Chehadeh et al., 1996; Fouard and Malandain, 2005; Sintorn and Borgefors, 2004), quadtree/octrees (Samet, 1990; Vörös, 2001), Face-Centered Cubic (FCC)/Body-Centered Cubic (BCC) grids (Fouard et al., 2007), etc. But, to our knowledge, no generic DT process has been designed on every kinds of image representations in two or higher dimensions.

This article deals with generalizing the DT computation on irregular isothetic grids (or i-grid for short). In two dimensions (2D), an i-grid is defined by a partition of ℝ² with rectangular cells whose edges are aligned along the two axes. The quadtree decomposition and the RLE (Run Length Encoding) are examples of classical techniques in imagery which induce an i-grid. Here, we focus our interest on generalizing techniques that compute the E²DT (squared Euclidean DT) of a d-dimensional (d-D) binary image (Coeurjolly and Montanvert, 2007; Maurer et al., 2003; Saito and Toriwaki, 1994). This transformation is a very common way to analyze the shape of graphical objects for various applications (see Paglieroni, 1992 and references of Maurer et al. (2003), for more details). Many of those techniques can be linked to the computation of a discrete Voronoi diagram of the background pixels (Coeurjolly and Montanvert, 2007; Hesselink et al., 2005). A further application of our work would be to compute E²DT in octree-based images, which are a common way to model very large-scale volumes (Samet, 1990).

In (Vacavant et al., 2008; Vacavant et al., 2009a), we introduced two kinds of extensions of the E²DT on i-grids. The first one is based on the distance between cell centres, while the second one computes the distance between a cell centre and the nearest foreground/background frontier. In this article, we make a complete review of each model, and we list their differences. We also recall previous work and present several techniques to compute them. In this review, we show that no efficient algorithm has been designed to compute both distance transformations in arbitrary dimensions. The methods that we introduce in this article are separable, i.e. they perform operations independently along the two axes and can be naturally extended to handle d-D images. Moreover, they can compute both E²DT extensions we have introduced in our previous work.

This paper is organized as follows: We first recall in Section 2 previous work about DT on the i-grid model, the two extensions of the E²DT on these grids, with their advantages and drawbacks. Then, we present the mathematical and algorithmic tools we need to develop separable DT processes on i-grid (Section 3). In Section 4, we describe two separable DT algorithms and we finally compare them (in terms of complexity and execution times), in Section 5.
2. Previous work about DT on I-grids

In this section, we first recall the concept of irregular isothetic grids (I-grids) in 2-D, with the following definition (Coeurejolly and Zerarga, 2006):

**Definition 1 (2-D I-grid).** Let \( P \) be a closed rectangular subset of \( \mathbb{R}^2 \). A 2-D I-grid \( G \) is a tiling of \( P \) with non overlapping rectangular cells whose edges are parallel to the \( X- \) and \( Y- \)axis. The position of each cell is characterized by its centre point (its position \((x_k,y_k) \in \mathbb{R}^2\) and its length along \(X-\) and \(Y-\)axis \((f_k^x, f_k^y) \in \mathbb{R}_{+}^2\)).

We consider this definition of 2-D I-grids for the rest of the paper, and we will shortly show that our contribution is easily extensible to the d-D case. In our framework, we consider labeled I-grids, i.e. each cell of the grid has a foreground or background label (its value is respectively “0” or “1” for example). For an I-grid \( G \), we denote by \( G_{B} \) and \( G_{F} \) the sets of foreground and background cells.

Here, we first recall the two extensions of the \( E^2 \)DT proposed on I-grids, for each cell \( R \in G_{F} \):

\[
\text{1-CDT}(R) = \min_{p} \left\{ d^2_{E^2}(p, \mathcal{P}) : R \in G_{B} \right\},
\]

\[
\text{1-BDT}(R) = \min_{s} \left\{ d^2_{E^2}(p, s) : s \in \mathcal{S} \right\},
\]

where \( \mathcal{S} \) is the set of segments contained in the background/foreground frontier of the grid. The first extension, the Centre-based DT (Eq. (1)) (Vacavant et al., 2008) is based on the cell centres of the grid, where \( d^2_{E^2}(p) \) is the squared Euclidean distance between the centres \( p = (x_k,y_k) \) and \( p' = (x_{k'},y_{k'}) \). The Border-based DT (Eq. (2)) (Vacavant et al., 2009a) employs cell borders, and computes the shortest distance between a foreground cell centre and the background/foreground boundary. We sum up in Table 1 the advantages and the drawbacks of each model.

As mentioned in Table 1, we can compute these extensions thanks to the construction of two kinds of Voronoi diagrams (VD). We recall that the VD of a set of points \( \mathcal{P} = \{p_i\} \) is a tiling of the plane into Voronoi cells (or VD cells) \( \{G_p\} \) (de Berg et al., 2000; Voronoi, 1908). The 1-D DT may be computed with such a structure, as we discussed in (Vacavant et al., 2008). If we now consider the background/foreground frontier to compute the 1-BDT, Eq. (2) implies that we compute a VD of segments, and not a classical VD of points (see Fig. 1 for an example of these diagrams computed on a simple I-grid). Hence, a simple approach to compute the 1-CDT (presented in Algorithm 1) is to pre-compute the complete VD of the background points (Vacavant et al., 2008), and to locate foreground points in the VD. In this case, the 1-CDT computation has a \( O(n \log n) \) time complexity, where \( n \) is the total number of cells, and \( n_B \) is the number of background cells. This technique is obviously not computationally efficient for every grids, and not adapted to dense grids (Vacavant et al., 2008).

To compute the 1-BDT, a similar process can be drawn, where the computation of the VD of segments can be handled in \( O(n_0 \log n_0) \), where \( n_0 = 4n_B \) is the total number of segments belonging to the background/foreground frontier (Karavelas, 2006). The extension of those transformations to d-D I-grids is difficult. A VD can be computed in d-D with a \( O(n_D \log n_D + n_D^2 \log n_D) \) time complexity (thanks to a gift-wrapping approach (de Berg et al., 2000) for example). However, localizing a point in the VD is an arduous task, and an additional structure like subdivision grids (Park et al., 2005) should be constructed to handle this operation. Hence, the building of the entire VD to obtain the 1-BDT is neither computationally efficient for every I-grids, nor easily extensible to higher dimensions.

**Algorithm 1. 1-CDT with the complete VD**

- **Input:** a labelled I-grid \( G \).
- **Output:** the 1-CDT of \( G \).

1. Compute the Voronoi Diagram \( \mathcal{V} \) of the points \( \{p ; R \in G_B\} \).
2. **foreach** cell \( R \in G_{F} \) do
   3. **locate** \( p \) in \( \mathcal{V} \);
   4. **if** \( p \) is the Voronoi vertex \( v \) of \( \mathcal{V} \) **then**
      5. \( s \leftarrow \) Voronoi site of an adjacent cell of \( v \);
   6. **else** \( p \) belongs to a Voronoi edge \( e \) in \( \mathcal{V} \) **then**
      7. \( s \leftarrow \) Voronoi site of an adjacent cell of \( e \);
   8. **else** \( (p \) belongs to a Voronoi cell \( c \) of \( \mathcal{V} \))
      9. \( s \leftarrow \) Voronoi site of \( c \);
   10. \( 1-CDT(R) \leftarrow d^2(E^2(s, p)) \);
3. **foreach** cell \( R \in G_{B} \) do 1-CDT \( (R) \leftarrow 0 \).

In Table 1, we also mention that one of the major problem of 1-CDT is that it is strongly dependent of the background representation (Vacavant et al., 2009a). In Fig. 2, we present an example of the computation of the 1-CDT of two I-grids where only the background cells differ. Since this definition is based on the cell centres position, the 1-CDT do not lead to the same distance map depending on the background encoding.

In a recent work (Vacavant, 2010), we have presented an efficient algorithm that permits to compute the 1-CDT with a sweepline based approach in 2-D (Breu et al., 1995). However, this technique is not easily extensible neither to higher dimensions, nor to 1-BDT. As a new contribution, we now propose two separable algorithms that compute both 1-CDT and 1-BDT that can be extended to the d-D case.

### 3. Separable minimization processes for 1-CDT and 1-BDT
#### 3.1. Separable \( E^2 \)DT algorithms on regular grids

The separable algorithm introduced by Saito and Toriwaki (1994) (one phase by dimension) first computes a one-dimensional (1-D) transformation along X-axis for each row of a binary image \( F \) of size \( n \times n \), which leads to a new image \( G = \{g(i,j)\} \) (see Fig. 3a). Then, a minimisation process is realized along Y-axis to obtain the final image \( H = \{h(i,j)\} \) (Fig. 3b): \( h(i,j) = \min\{g(i,j)^2 + (j - y)^2; 1 \leq y \leq n\} \).

The main idea of the separable method of Maurer et al. (2003) is that the intersection between the complete VD of background pixels (i.e. sites) and a line of the grid can be easily computed, then simplified. Indeed, for a row \( j \), the VD sites can be “deleted” by respecting three remarks mainly based on the monotonicity of the \( d_F \) distance (Maurer et al., 2003): (1) if we consider the line

<table>
<thead>
<tr>
<th>I-CDT</th>
<th>I-BDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Based on distance between cells centres</td>
<td>Based on centre-to-border distance</td>
</tr>
<tr>
<td>Can be computed thanks to the construction of a Voronoi diagram of points</td>
<td>Can be computed thanks to the construction of a Voronoi diagram of segments</td>
</tr>
<tr>
<td>* Is extensible to other non-isothetic grids (hexagonal, triangular, etc.)</td>
<td>* Not easily extensible to these grids</td>
</tr>
<tr>
<td>* Is not independent from background encoding</td>
<td>* The result does not depend on the background representation</td>
</tr>
<tr>
<td>* Can be computed in 2-D (but not in d-D) in linear time with a sweep line algorithm (Vacavant, 2010)</td>
<td>* Not easily computable by this algorithm</td>
</tr>
</tbody>
</table>

This algorithm requires the computation of the complete VD of segments to compute the 1-CDT, which is computationally expensive. In contrast, the 1-BDT can be computed in a more efficient manner by using a sweepline approach, as described in Algorithm 1.

In Table 1, we also mention that one of the major problem of 1-CDT is that it is strongly dependent of the background representation (Vacavant et al., 2009a). In Fig. 2, we present an example of the computation of the 1-CDT of two I-grids where only the background cells differ. Since this definition is based on the cell centres position, the 1-CDT do not lead to the same distance map depending on the background encoding.

In a recent work (Vacavant, 2010), we have presented an efficient algorithm that permits to compute the 1-CDT with a sweepline based approach in 2-D (Breu et al., 1995). However, this technique is not easily extensible neither to higher dimensions, nor to 1-BDT. As a new contribution, we now propose two separable algorithms that compute both 1-CDT and 1-BDT that can be extended to the d-D case.
If \( y = j \in \mathbb{Z} \), then we only have to keep the nearest VD sites from \( L(2) \); those sites can be ordered along the X-axis; which means that it is not necessary to keep the complete VD, but only its intersection with the line \( L(3) \). A VD site may be hidden by two neighbour sites, and thus not considered anymore (Fig. 4b). In this article, we show the extension of this technique on I-grids by adapting these properties on these grids.

### 3.2. Separable computation of I-CDT and I-BDT

To develop a separable process on I-grids, we use the irregular matrix \( M \) associated to a labeled I-grid \( G \) introduced in (Vacavant et al., 2008). This data structure aims to organize the cells of the grid along the X- and Y-axis by adding virtual cells centres (see Fig. 5 for an example). We show in Section 5 that, even if we add points in this structure, separable processes on the irregular matrix are faster than the complete VD based approach we presented in the previous section (Algorithm 1). More precisely, the irregular matrix contains as many columns (respectively rows) as X-coordinates (Y-coordinates) in the grid. These coordinates are stored in two tables \( T_X \) and \( T_Y \) (and we denote \( n_1 = |T_X| \) and \( n_2 = |T_Y| \)). At the intersection of two X- and Y-coordinates, a node in \( M \) has the same value as the cell containing it, and may represent the cell centre or not (i.e., this is an extra node, see Fig. 5a). The extra nodes are used.

![Figure 1: Example of the VD of the background points to obtain the I-CDT of a simple I-grid (a) and the background/foreground frontier (dark segments) to obtain the I-BDT (b).](image1)

![Figure 2: The result of the I-CDT of the complete regular grid (b) computed from the binary image (a) and an I-grid where the foreground is regular and the background is encoded with a RLE scheme along Y (c). The distance value \( d \) of a cell is represented with a grey level \( \frac{c}{255} \). The contour of the object (background/foreground frontier) is recalled in each distance map (b) and (c) with a smooth curve. We also present the associated elevation map of each distance map in (d) and (e).](image2)

![Figure 3: Double minimization process on a small binary image. The first step of the algorithm computes \( G \) (a), representing minimal distances along X axis. If we choose one column of \( G \), the second phase consists in computing \( g(i, j) + (j - y)^2 \) for each pixel of this column (b). The minimization of this column lead to the computation of the E^D(T) stored in \( H \).](image3)
to propagate the distance values through the irregular matrix and then compute a correct distance value for each cell. To apply the \(t\)-BDT on this data structure, we also have to take into account the border and the size of the treated cells. We thus add for each node \(\mathcal{M}(i, j)\) border attributes along \(X\)- and \(Y\)-axis that permit to propagate the distance values to cell borders through \(\mathcal{M}\). For a node \(\mathcal{M}(i, j)\) contained by the cell \(R\) in \(G\), we denote respectively by \(H(i, j)\) and \(H_{\alpha}(i, j)\) the attributes that represent the minimum between the distance to the left (respectively right) border of \(R\) along \(X\), and the distance to the neighbour node at the left (right) position in \(\mathcal{M}\). In the same manner, we define \(H_{\alpha}(i, j)\) and \(H_{\alpha}(i, j)\) in respect to the top (bottom) border of \(R\) and neighbour nodes at the top (bottom) position in \(\mathcal{M}\) (see Fig. 5b). Building the irregular matrix of an \(i\)-grid \(G\) can be handled in \(O(n_1n_2)\) time complexity. More precisely, we first scan all the cells of \(G\) to know the \(n_1\) columns and the \(n_2\) rows of \(\mathcal{M}\). Then, we consider each node of \(\mathcal{M}\) and we assign its background or foreground value and its border attributes.

We now present how to adapt Eq. (1) on the irregular matrix thanks to a separable minimization process:

**Proposition 1** (Separable \(t\)-CDT). Let \(\mathcal{M}\) be the associated irregular matrix of the 2-D \(i\)-grid \(G\). Then the \(t\)-CDT of \(G\) can be decomposed in two separable processes, and consists in computing the matrix \(\mathcal{M}'\) and \(\mathcal{M}''\) as follows:

\[
\mathcal{M}'(i, j) = \min_x \{ |T_X(i) - T_X(x)| : x \in \{0, \ldots, n_1 - 1\} \text{ and } \mathcal{M}(x, j) = 0 \}, \\
\mathcal{M}''(i, j) = \min_{y} \{ \mathcal{F}(j) : y \in \{0, \ldots, n_2 - 1\} \},
\]

where \(\mathcal{F}(j)\) is a parabola given by:

\[
\mathcal{F}(j) = M'(i, y)^2 + (T_Y(j) - T_Y(y))^2.
\]

In this case, this transformation is equivalent to a minimization process of 2-D quadratic forms. Eq. (2) can also be adapted on the irregular matrix with a minimization process along the two axes:

**Proposition 2** (Separable \(t\)-BDT). Let \(\mathcal{M}\) be the associated irregular matrix of the 2-D \(i\)-grid \(G\). Then the \(t\)-BDT of \(G\) can be decomposed in two separable processes, and consists in computing the matrix \(\mathcal{M}'\) and \(\mathcal{M}''\) as follows:

\[
\mathcal{M}'(i, j) = \min_{x} \{ |T_X(i) - T_X(x)| : x \in \{0, \ldots, n_1 - 1\} \text{ and } \mathcal{M}(x, j) = 0 \}, \\
\mathcal{M}''(i, j) = \min_{y} \{ \mathcal{G}(j) : y \in \{0, \ldots, n_2 - 1\} \},
\]

where \(\mathcal{G}(j)\) is a flattened parabola given by:

\[
\mathcal{G}(j) = \begin{cases} 
M'(i, y)^2 + (T_Y(j) - T_Y(y) - H_{\alpha}(i, y))^2 & \text{if } T_Y(j) - T_Y(y) > H_{\alpha}(i, y), \\
M'(i, y)^2 + (T_Y(j) - T_Y(y) - H_{\alpha}(i, y))^2 & \text{if } T_Y(j) - T_Y(y) < H_{\alpha}(i, y), \\
M'(i, y)^2 & \text{otherwise}.
\end{cases}
\]

We prove in (Vacavant et al., 2009b) that this proposition is correct, i.e. this two-dimensional scheme consists in computing the

\(\text{Fig. 4.}\) In a regular grid, when we treat a row \(l\) with Maurer et al. algorithm, we consider the VD of background nodes like in (a). We only keep the nearest VD sites of \(l\) and we obtain (c) by deleting sites which associated VD cells intersect the \(l\) row. Arrows indicate the associated VD site of each foreground pixel of this row.

\(\text{Fig. 5.}\) Construction of the irregular matrix associated to the simple \(i\)-grid illustrated in Fig. 1. In (a), the extra node \(\mathcal{M}(0, 2)\) of the matrix is depicted at the intersection of the dotted lines. In (b), examples of values for border attributes are also given along \(X\) (dashed lines) and \(Y\) (dotted lines). For instance, we have \(H_L(0, 2) = H_R(0, 2) = 20\), \(H_T(0, 2) = H_B(0, 2) = 10\), \(H_{\alpha}(1, 1) = H_{\alpha}(1, 1) = 10\), and \(H_L(1, 1) = H_R(1, 1) = 0\) since this node coincides with a cell horizontal border. We can also notice that \(H_{\alpha}(2, 3) = H_{\alpha}(2, 3) = 5\).
t-BDT given in Eq. (2). One can notice that it is possible to adapt the proof of this proposition to prove Proposition 1. This separable minimization process also permits to compute the t-CDT (Eq. (1)), if we assign all border attributes to zero. Here, a flattened parabola (see also Fig. 6) is an ad hoc function composed by two half-parabolas and a constant function. It represents a constant distance value from a node \( M(i,j) \) to its neighbour nodes and cell borders, and then increases as a classical quadratic function beyond those limits. The computation of matrix \( M^* \) in Proposition 2 thus consists in computing the lower envelope of a set of flattened parabolas. This operation is possible since these functions are convex (composition of convex functions), and there exist a single intersection or an infinity of intersections between two flattened parabolas. In the regular case, \( I^2 \)DT algorithms based on a 2-D quadratic form minimization (Hirata, 1996; Meijster et al., 2000; Saito and Toriwaki, 1994) are linear and correct, since the intersection between parabolas to realize a correct minimization scheme of a 2-D quadratic form, conform to Propositions 1 and 2. The function \( CSep^2(u,v) \) is the exact coordinate of the

\[
g_o(u,v) = \begin{cases} 
  g_o^2 + (u - y - lu_1)^2 & \text{if } v - y > lv_1, \\
  g_o^2 + (y - u - lu_2)^2 & \text{if } y - u > hu_2, \\
  g_o^2 & \text{otherwise}
\end{cases}
\]

where:

- \( g_o, g_v, lu_1, lu_2, lv_1, lv_2, u \) and \( v \) are positive or null real numbers;
- \( \text{if } u > v \geq 0, \text{then } u - lu_1 \geq v + lv_2; \)
- \( \text{if } 0 \leq u \leq v, \text{then } u + lu_2 \leq v - lu_1. \)

The intersections between these parabolas is either reduced to a point, or reduced to an infinite set of points.

Notice that the elements \( M^*(i,y), H_B(i,y), H_T(i,y) \) and \( T_y(y) \) of \( G_y \) in Proposition 2 respectively correspond to \( g_o, lu_1, hu_2, \) and \( g_v \). They also respect the conditions given at the end of Lemma 1. For example, let \( G_1 \) and \( G_2 \) be a two flattened parabolas such that \( y_1 \geq y_2 \), we can easily verify that \( y_1 - H_B(i,y_1) \geq y_2 + H_T(i,y_2) \) (see Figs. 5 and 6). The proof of this lemma and a way to compute a valid intersection point are given in (Vacavant et al., 2009b).

4. Proposed separable methods to compute ICDT and t-BDT

In this section, we first present the adaptation of Saito et al. algorithm (Saito and Toriwaki, 1994), because this is a natural application of the ICDT and t-BDT minimization processes we described in Propositions 1 and 2. Then, we show how to use Maurer et al. approach (Maurer et al., 2003) for the same purpose, which lead to the development of a faster algorithm.

4.1. Adaptation of Saito et al. \( I^2 \)DT Algorithm on \( \mathbb{Z} \)-grids

Algorithm 2. Separable t-BDT inspired from Saito and Toriwaki (1994) (1/2)

\[
\begin{align*}
\text{Algorithm 2. Separable t-BDT inspired from Saito and Toriwaki (1994) (1/2)} \\
\text{input: A labeled } \mathbb{Z}\text{-grid } G. \\
\text{output: The t-BDT of } G, \text{ stored in an irregular matrix } M^*. \\
1 & \text{build the irregular matrix } M^* \text{ associated to } G; \\
2 & \text{for } j = 0 \text{ to } n_2 - 1 \text{ do } \text{(First stage along } X) \\
3 & \quad \text{if } M(0,j) = 0 \text{ then } M^*(0,j) \leftarrow 0; \\
4 & \quad \text{else } M^*(0,j) \leftarrow \infty ; \\
5 & \quad \text{for } i = 1 \text{ to } n_1 - 1 \text{ do } \\
6 & \quad \quad \text{if } M(i,0) = 0 \text{ then } M^*(i,0) \leftarrow 0; \\
7 & \quad \quad \text{else} \\
8 & \quad \quad \quad \text{if } M^*(i-1,j) = 0 \text{ then} \\
9 & \quad \quad \quad \quad M^*(i,j) \leftarrow T_B(i) - T_B(i-1) - H_B(i-1,j); \\
10 & \quad \quad \quad \text{else} \\
11 & \quad \quad \quad \quad M^*(i,j) \leftarrow T_B(i) - T_B(i-1) + M^*(i-1,j); \\
12 & \quad \quad \text{for } i = n_1 - 2 \text{ to } 0 \text{ do } \\
13 & \quad \quad \quad \text{if } M^*(i+1,j) < M^*(i,j) \text{ then} \\
14 & \quad \quad \quad \quad M^*(i,j) \leftarrow T_B(i+1) - T_B(i) - H_B(i,j); \\
15 & \quad \quad \quad \text{else} \\
16 & \quad \quad \quad \quad M^*(i,j) \leftarrow T_B(i+1) - T_B(i) + M^*(i-1,j); \\
\end{align*}
\]

In this section, we present the complete algorithm inspired from Saito and Toriwaki (1994) (Algorithm 2, divided into two main parts), and how to modify it in order to compute the ICDT and t-BDT. As we discussed in the previous section, we need to compute the intersection between parabolas to realize a correct minimization scheme of a 2-D quadratic form, conform to Propositions 1 and 2. The function \( CSep^2(u,v) \) is the exact coordinate of the

![Fig. 6: Important features of a flattened parabola, composition of two half-parabolas (in dotted lines) and a constant function.](image-url)
intersection point between two classical parabolas (Coeurjolly and Montanvert, 2007; Meijster et al., 2000): 

\[ M'(i, u)^2 - M'(i, v)^2 = \frac{-M(i, u)^2 + M(i, v)^2}{2(T_Y(v) - T_Y(v))}. \]

For I-BDT, we introduce the \( BSep(u, v) \) function thanks to the proof of Lemma 1 (see Vacavant et al., 2009b).

Algorithm 3. Separable I-BDT inspired from Saito and Toriwaki (1994a)/(2/2)

```
input: A labeled I-grid \( G \).
output: The I-BDT of \( G \), stored in an irregular matrix \( M' \).
1 for \( i = 0 \) to \( n_1 - 1 \) do \hfill \{Second stage along \( Y \)\}
2 \( q \leftarrow 0 \); \( x[0] \leftarrow 0 \); \( y[0] \leftarrow 0 \);
3 \( h[q] \leftarrow H_Y(x[q]); \) \( h[q] \leftarrow H_Y(y[q]); \)
4 for \( j = 1 \) to \( n_2 - 1 \) do \hfill \{\( CSep \) function for I-CDT\}
5 \( q \leftarrow q - 1 \);
6 if \( q < 0 \) then \hfill \{Process same first stage along \( X \) as Algorithm 2\}
7 \( q \leftarrow 0 \); \( s[q] \leftarrow j \);
8 \( h[q] \leftarrow H_x(s[q]); \) \( h[q] \leftarrow H_x(j); \)
9 else \hfill \{\( CSep \) function for I-CDT\}
10 \( w \leftarrow BSep(s[q]); \)
11 if \( w < \tau(n_1 - 1) \) then \hfill \{Process same first stage along \( X \) as Algorithm 2\}
12 find the node \( M(i, k) \), \( k \in \{s[q], ..., n_2 - 1\} \) such that \( T_Y(k) > w \);
13 \( q \leftarrow q + 1 \); \( s[q] \leftarrow k \); \( t[q] \leftarrow w \);
14 \( h[q] \leftarrow H_x(s[q]); \) \( h[q] \leftarrow H_x(t[q]); \)
15 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
16 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
17 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
18 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
19 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
```

```
This function is defined by:

\[
\begin{align*}
\bar{y} &= T_Y(v) - H_x(i, v) - \sqrt{M'(i, u)^2 - M'(i, v)^2}, \\
BSep(u, v) &= \begin{cases} \\
\hat{y} & \text{if } T_Y(u) - H_x(i, u) < \hat{y} \leq T_Y(u) + H_x(i, u), \\
\text{CSep}(T_Y(u) - H_x(i, u), T_Y(u) + H_x(i, u)) & \text{if } \hat{y} < T_Y(u) - H_x(i, u) \leq T_Y(u) + H_x(i, u), \\
\text{CSep}(T_Y(u) + H_x(i, u), T_Y(u) - H_x(i, u)) & \text{if } T_Y(u) - H_x(i, u) \leq T_Y(u) + H_x(i, u) \leq \hat{y}. 
\end{cases}
\end{align*}
\]

In the first step, we can notice that the only difference with the regular square case (Coeurjolly and Montanvert, 2007; Saito and Toriwaki, 1994a) is the computation of the distance, lines 2 to 17 (part 1/2). We have to consider in those operations the distance between the point \( M(i, j) \) and its neighbors. This is indeed a double linear scan for each row, where we use \( H_x \) and \( H_y \) attributes to propagate the distance to cells borders.\(^1\) During the second phase, we use stacks \( s \) and \( t \), as in (Meijster et al., 2000), and two additional stacks \( h_r \) and \( h_b \) to propagate border distances through the columns of the matrix. This step is equivalent to the computation of the lower envelope of a set of flattened parabolas (see Fig. 7). The computation of \( w \) (line 11 part 2/2) only depends on the function \( BSep(u, v) \) and then permits to find the intersection point in \( M' \) between two flattened parabolas (line 13 part 2/2). It is replaced by \( CSep(u, v) \) in the case of the I-CDT. The find command is performed with a dichotomous search through the ordered set of nodes \( \{M(i, k)\}_{0 \leq |k| \leq n_2 - 1} \), and has a \( O(\log n_2) \) time complexity in the worst case. But in our experiments (see Section 5), we have observed that this is a fast operation, since we begin the search from the last intersection point (with index \( s[q] \)).

4.2. Adaptation of R. Maurer et al. E²DT Algorithm on I-grids

The first stage of this method (Algorithm 3) consists in scanning along \( X \) and in initializing the distance of each node of the irregular matrix, as in Algorithm 2. In the second part of our algorithm, we build a partial VD intersected with each column \( i \). As in the original algorithm (Maurer et al., 2003), we use two stacks storing real successive coordinates of treated sites \( (h, \) and their squared distance \( g \). The first loop of this function (line 7) corresponds to the deletion of hidden sites, thanks to the hidden_by predicate.


```
input: the labeled I-grid \( G \).
output: the I-BDT of \( G \), stored in the irregular matrix \( M' \).
1 \( i = 0 \) to \( n_1 - 1 \) do \hfill \{Second stage along \( Y \)\}
2 \( l = 0 \); \( g = \emptyset \); \( h = \emptyset \);
3 \( f_T \leftarrow h_f \leftarrow g_f \leftarrow \emptyset \);
4 \( j = 0 \) to \( n_2 - 1 \) do \hfill \{\( CSep \) function for I-CDT\}
5 \( l \leftarrow 0 \); \( g \leftarrow \emptyset \); \( h \leftarrow \emptyset \);
6 \( f_T \leftarrow h_f \leftarrow g_f \leftarrow \emptyset \);
7 if \( M'(i, j) \neq \infty \) then \hfill \{Process same first stage along \( X \) as Algorithm 2\}
8 \( \text{while } l > 2 \text{ and hidden\_by} \)
9 \( l \leftarrow l + 1 ; \)
10 \( l \leftarrow l + 1 ; \)
11 \( f_T \leftarrow H_x(i, j); \)
12 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
13 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
14 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
15 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
16 \hfill \{Function hidden\_by\} \hfill \{Function hidden\_by\}
```

```
Function hidden\_by
```

```
input: Y-coordinates of three points in \( \mathbb{R}^2 \)
denoted by \( y_p, y_q, y_w \), and their squared distance to the line \( L: y = r \) denoted by \( d_y^2(u, L), d_y^2(v, L), d_y^2(w, L) \).
output: is \( v \) hidden by \( u \) and \( w \)?
1 \( a \leftarrow y_q - y_p \); \( b \leftarrow y_w - y_q \); \( c \leftarrow a + b \)
2 return \( c \times d_y^2(u, L) - b \times d_y^2(v, L) - a \times d_y^2(w, L) - abc > 0; \)
```

In Algorithm 3, we also use two additional stacks, denoted by \( f_T \) and \( g_f \) to store the border attributes and update them. Thanks to these stacks, the second stage of our algorithm is achieved in linear time. By testing the value of \( l \) (line 12), we know if we have to scan again the stacks and to update the distance values of the nodes. Finally, we linearly scan the stacks to find the nearest border of the \( M(i, j) \) current node (line 15), and this step actually consists in considering the lower envelope of a set of flattened parabolas, as in Algorithm 2 (see also Fig. 7).

4.3. Complexity analysis

In Algorithms 2 and 3, the first operation (build the irregular matrix) is performed in \( O(n_1 n_2) \) time. More precisely, we first scan all the cells of \( G \) to get the \( n_1 \) rows and \( n_2 \) columns of \( M \). Algorithm

\(^1\) If we consider an I-CDT process, we can set these values to zero, or use the complete description given in (Vacavant et al., 2008).
2 has a global time complexity in $O(n_1^2 \log n_2^2)$. It can be easily extended to higher dimensions: the step 1 stands as an initialization step, and for each greater dimension, a mixing process, as step 2, permits to combine results obtained in the lower dimensions.

For a labeled $d$-D $I$-grid, which associated irregular matrix size is $n_1, \ldots, n_d$, the cost of the consecutive steps is in $O(n_1 \ldots n_d \log n_1 \ldots \log n_{d-1})$.

Algorithm 3 realizes a linear $I$-BDT algorithm in respect to the associated irregular matrix size, i.e. in $O(n_1 \log n_2)$ time complexity. As Algorithm 2, this method is easily extensible to higher dimensions: we still realize the first step as an initialization phase, and for each dimension $d > 1$, we combine results obtained in dimension $d - 1$. If we consider a labeled $d$-D $I$-grid, which associated irregular matrix size is $n_1, \ldots, n_d$, the time complexity of our algorithm is thus in $O(n_1 \ldots n_d)$.

In the next section, we present experiments to show the interest of the $I$-BDT and $I$-CDT, and to point out that the last algorithm is a very efficient approach to compute both $I$-CDT and $I$-BDT of various classical $I$-grids.

5. Experimental analysis

Here, we first propose to study the computation of the $I$-CDT on several classical $2$-D grids in imagery. We consider in our experiments three regular grids (square, rectangular and hexagonal), then two image-dependent grids, based on a quadtree decomposition and a RLE along $Y$ scheme. Since the definition of the $I$-CDT is only based on the centres of the cells, we can treat all those grids that belong to the $I$-grid model, and other non-isothetic grids, like hexagonal or triangular grids. We
present the computation results of the $I$-CDT for those grids, based on the small image "ghost" (16 × 16 pixels) in Fig. 9. In this figure, the rectangular grid cells have a height two times greater than the width. We recall that $I$-CDT is an extension of $E^2$DT conform to classical $E^2$DT definitions on regular grids, and may be extended to such $d$-D grids.

We now propose to present the result of our $I$-BDT algorithm for the binary image depicted at the beginning of this article in Fig. 2, digitized in various classical $I$-grids used in imagery. Fig. 10 illustrates $I$-BDT elevation maps where the background of the original image is represented with a regular square grid ($D$), a quadtree ($qT$) and a RLE along $Y(L)$. The foreground is encoded with an RLE along $Y$-axis. We can notice in this figure that the result of the $I$-BDT is independent of the representation of the background. The distance values sin the foreground region are thus the same in the three elevation maps of a given column.

We now focus our interest on the execution time of the three algorithms presented in this article. Moreover, we take into account the $I$-CDT 2-D approach that we introduced in (Vacavant, 2010). Hence, we consider the four algorithms presented in Table 2. The first algorithm represents the simple approach we discussed in Section 2, which is hardly extensible to $d$-D treatments, and $I$-BDT computation. The fourth technique was recently introduced in (Vacavant, 2010) and is based on a sweep-line process (Breu et al., 1995). For this algorithm, extension to $I$-BDT and $d$-D case are also difficult. As its complexity is quasi optimal, it represents a reference for our experiments. Algorithm 2 is inspired from the quadratic form minimization scheme of Saito and Toriwaki (1994). Algorithm 3 is the separable transformation inspired from Maurer et al. (2003). In Fig. 8, we present the three chosen images for our experiments, and in Table 3, we show the execution times for these four algorithms, for $I$-grids built from three binary images. We have performed those experiments on a mobile workstation with a 2.2 GHz Intel Centrino Duo processor, and 2 GB sRAM. We can first notice that Algorithm 3 gives good results for $I$-CDT, even compared to the optimal Algorithm 4. Indeed, this is the fastest one for regular square $I$-grids, and is very competitive for sparse $I$-grids (e.g. near one half second for $qT$ and $L$ based algorithms)

Table 2

<table>
<thead>
<tr>
<th>Id.</th>
<th>Algorithm</th>
<th>Time</th>
<th>Space</th>
<th>$d$-D</th>
<th>$I$-CDT</th>
<th>$I$-BDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Complete VD (Vacavant et al., 2008; Voronoi, 1908)</td>
<td>$O(n \log n)$</td>
<td>$O(n)$</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>From Vacavant et al. (2008) and Saito and Toriwaki (1994)</td>
<td>$O(n_1 n_2 \log n_2)$</td>
<td>$O(n_1 n_2)$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>From Vacavant et al. (2009a) and Maurer et al. (2003)</td>
<td>$O(n_1 n_2)$</td>
<td>$O(n_1 n_2)$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>From Vacavant (2010) and Breu et al. (1995)</td>
<td>$O(n)$ best, $O(n_1 n_2)$ worst</td>
<td>$O(n)$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
on image canon). In the latter case (canon), Algorithms 1 and 4 are faster than Algorithm 3, but we recall that they are hardly extensible to higher dimensions. Algorithm 2 is interesting for dense grids, but is naturally overtaken by Algorithm 3. For the l-grid, Algorithms 2 and 3 suffer from an execution time increase, mainly one second.

For the background/foreground frontier (l-CDT) and the l-BDT (b) versions, with competitive execution time and complexities.

### Table 3
We present execution times (in seconds) for each algorithm for the l-CDT (a) and the l-BDT (b) and for each l-grid. Number inside parenthesis in (b) are the increasing rate in % between l-CDT and l-BDT execution times.

<table>
<thead>
<tr>
<th>Image</th>
<th>Algorithm 1</th>
<th></th>
<th></th>
<th>Algorithm 2</th>
<th></th>
<th></th>
<th>Algorithm 3</th>
<th></th>
<th></th>
<th>Algorithm 4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D</td>
<td>q</td>
<td>L</td>
<td>D</td>
<td>q</td>
<td>L</td>
<td>D</td>
<td>q</td>
<td>L</td>
<td>D</td>
<td>q</td>
</tr>
<tr>
<td><strong>(a) l-CDT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise</td>
<td>0.255</td>
<td>0.104</td>
<td>0.053</td>
<td>Noise</td>
<td>0.037</td>
<td>0.077</td>
<td>0.044</td>
<td>Noise</td>
<td>0.046</td>
<td>0.065</td>
<td>0.038</td>
</tr>
<tr>
<td>Lena</td>
<td>1.413</td>
<td>0.145</td>
<td>0.081</td>
<td>Lena</td>
<td>0.192</td>
<td>0.376</td>
<td>0.245</td>
<td>Lena</td>
<td>0.185</td>
<td>0.166</td>
<td>0.135</td>
</tr>
<tr>
<td>Canon</td>
<td>36.236</td>
<td>0.273</td>
<td>0.234</td>
<td>Canon</td>
<td>1.678</td>
<td>1.322</td>
<td>1.316</td>
<td>Canon</td>
<td>1.134</td>
<td>0.485</td>
<td>0.585</td>
</tr>
<tr>
<td><strong>(b) l-BDT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise</td>
<td>0.047 (27)</td>
<td>0.100 (29)</td>
<td>0.054 (23)</td>
<td>Noise</td>
<td>0.065 (42)</td>
<td>0.085 (31)</td>
<td>0.049 (29)</td>
<td>Lena</td>
<td>0.258 (40)</td>
<td>0.170 (26)</td>
<td>0.170 (26)</td>
</tr>
<tr>
<td>Canon</td>
<td>2.248 (34)</td>
<td>2.107 (59)</td>
<td>2.020 (33)</td>
<td>Canon</td>
<td>1.507 (33)</td>
<td>0.563 (16)</td>
<td>0.718 (23)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. Conclusion and future works

In this article, we have recalled the two extensions of the E²DT on l-grids based on the cells centres (l-CDT) and on the background/foreground frontier (l-BDT). An application of the algorithms presented in this paper may be the computation of E²DT in octree-based images for example. These algorithms, inspired from Maurer et al. (2003) and Saito and Toriwaki (1994), are separable; i.e. they are easily extensible to higher dimensions. The last algorithm based on the work of Maurer et al. (2003) is able to efficiently compute both l-CDT and l-BDT thanks to the irregular matrix structure, with competitive execution time and complexities.

As a future work, we first would like to study E²DT on other generic image data structures, like (3-D and more generally d-D) point sets or triangulations for example. Moreover, we are working on developing efficient tools to compute the discrete Euclidean reduced medial axis transform (Coeurjolly and Montanvert, 2007) on l-grids. We plan to propose centred simple forms of d-D binary objects represented within various image structures.

### References


